Fondamenti di Informatica T-1 (A.A. 2013/2014) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Giovedì 10 Luglio 2014 – durata 1h
Totale 12 punti, sufficienza con 7

ESERCIZIO 1 (6 punti)

Dati un carattere `op` e due liste di interi `l1` e `l2` della stessa dimensione, si realizzi una funzione ITERATIVA

```
list fun(char op, list l1, list l2);
```

che restituisca una nuova lista di interi utilizzando `op` come operatore binario e gli elementi di `l1` e `l2` come operandi. In particolare, `op` può essere uno dei caratteri `'+'`, `'-'`, `'*'`, `'/'` rappresentanti rispettivamente addizione, sottrazione, moltiplicazione e divisione. La funzione `fun` deve prendere gli elementi di `l1` e `l2` a coppie (un elemento di `l1` e un elemento di `l2` alla volta), applicare l'operazione corretta in relazione all'operatore dato e restituire il risultato come elemento della nuova lista.

Per esempio, se op = `'+'`, `l1 = [5, 2]`, e `l2 = [9, 8]` la funzione `fun()` deve restituire la lista `[14, 10]` in quanto `5+9=14` e `2+8=10`.

Si definisca inoltre la funzione

```
int operate(char op, int a, int b);
```

che, dati in ingresso un carattere e due interi che rappresentano l'operatore e i due operandi, esegua l'operazione e ne restituisca il risultato intero. Se l'operatore passato come parametro d'ingresso non è tra quelli previsti, la funzione deve restituire 0.

La funzione `fun()` dovrà essere implementata utilizzando le primitive dell'ADT lista; ogni altra funzione, inclusa la funzione `operate()`, dovrà essere opportunamente specificata dal candidato. Si realizzi inoltre una semplice funzione `main()` di prova che invochi correttamente la funzione `fun()` creata.

Nota: l'ordine degli elementi della lista restituita dalla funzione `fun()` è ininfluente.

ESERCIZIO 2 (2 punti)

Un elaboratore rappresenta i numeri interi su 8 bit tramite la notazione in complemento a 2. Indicare come viene svolta la seguente operazione aritmetica calcolandone il risultato secondo la rappresentazione binaria in complemento a 2 (si trasli anche il risultato in decimale per verificare la correttezza dell'operazione):

- 63 + 82

ESERCIZIO 3 (3 punti)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int control(int a, int b, int c){
   int delta = b*b - (4*a*c);
   printf("delta: %d \n", delta);
   if(delta<=0)
      return 0;
   else
      return 1;
}

int fun(int *d1, int *d2, int *d3){
   int flag = 0;
   flag = control(*(++d3), *(++d2), *(++d1));
   while(flag == 0){
      flag = control(*(++d3), *(++d2), *(++d1));
   }
   return flag;
}

int main(void){
   int d1[5] = {1,2,-3,4,7};
   int d2[5] = {5,-2,3,8,9};
   int d3[5] = {9,10,2,-1,-2};

   int res = fun(d1, d2, d3);

   if(!res)
      printf("Success.\n");
   else
      printf("Fail.\n");

   return (0);
}
```

ESERCIZIO 4 (1 punto)

Il candidato illustri brevemente cosa sono e come vengono usate le strutture dati di tipo stack.

Soluzioni

ESERCIZIO 1

```
int operate(char op, int a, int b)
{
      switch (op) {
          case '+':
                  return a+b;
                  break;
          case '-':
                  return a-b;
                  break;
          case '*':
                  return a*b;
                  break;
          case '/':
                  return a/b;
                  break;
          default:
                  return 0;
                  break;
          }
}

list fun(char op, list l1, list l2){
      int a,b;
      list res = emptylist();

      while(!empty(l1)){

            a = head(l1);
            b = head(l2);

            res = cons(operate(op, a, b), res);

            l1 = tail(l1);
            l2 = tail(l2);
      }

      return res;
}

int main(){
      char op = '+';
      list l1 = cons(5,cons(2, emptylist()));
      list l2 = cons(9,cons(8, emptylist()));

      list res = fun(op, l1, l2);

      while(!empty(res)){
            printf("%d \n", head(res));
            res = tail(res);
      }

      return 0;
}
```

ESERCIZIO 2

- 63 + 82 = 19

+ 63 → 00111111 = 32+16+8+4+2+1
  11000000
- 63 → 11000001

+ 82 → 01010010 = 64+16+2
+ 19 → 00010011 = 16+2+1

ESERCIZIO 3

Il programma è corretto sintatticamente e la sua esecuzione produce la stampa:

```
delta: -76
delta: 33
Fail.
```

La funzione `main()` inizializza tre array di interi `d1`, `d2` e `d3` e poi invoca la funzione `fun()`.

La funzione `fun()` invoca subito la funzione `control()` a cui passa i valori degli elementi puntati da `d3`, `d2` e `d1`, spostando i puntatori alla seconda posizione. Una volta riacquisito il controllo, la funzione `fun()` entra in un ciclo `while` in cui rimarrà fintanto che il valore di `flag` sarà pari a 0. Ad ogni iterazione viene richiamata la funzione `control()`alla quale sono passati i valori attualmente puntati da `d3`, `d2` e `d1`, dopo aver spostato i puntatori alla posizione successiva. Se il valore di `flag` è ancora nullo, si passa all'iterazione successiva, altrimenti si esce dal ciclo `while`.
Infine la funzione `fun()`restituisce al chiamante il valore della variabile intera `flag`.

La funzione `control()` calcola la formula del delta (`b^2 - 4ac`) in base ai parametri ricevuti in ingresso e stampa sullo standard output il suo valore. Se il delta è minore o uguale a zero, la funzione ritornerà il valore 0, altrimenti 1.

La funzione `main()` stampa sullo standard output la stringa `"Success."` nel caso in cui il valore di `!res` sia pari a 0, altrimenti la stringa `"Fail."`.